Powering the Next Generation of Smart Devices

By Kristopher Ardis, Executive Director, Micros, Security & Software Business Unit

April 2018

maxim integrated™

Abstract

As the things in our everyday lives become smarter, design engineers must find practical ways to address the power demands of these devices. When considering internet of things (IoT) applications, however, power considerations are often left to the end of the design cycle. This paper examines the power considerations of these types of applications, as well as the role of low-power microcontrollers in efficiently powering connected devices.

Introduction

How can we power all of our invisibly intelligent devices?

Things have intelligence

I'm not suggesting that your front door has a brain. But if your front door had a way to talk, it could have plenty of things to say. It could tell you if the door was open or closed, locked or unlocked. It could tell you what the weather was like just outside. It could tell you who was at the door. And if your door had a way to listen to you, it could do even more: lock or unlock the door per your command, let you communicate with that person waiting at the door.

The things we have around us and interact with have intelligence. That's why we want to connect them to the internet. But many of these things don't currently have a way to share their knowledge or listen to our requests—the technology isn't there to make it worthwhile or practical to connect everything that might have valuable data. The sensors and actuators that can bring intelligence to the things around us have formidable power challenges to overcome. Running power to a front-door sensor will limit adoption. Frequent battery replacements will end up in unused devices (and unused intelligence). Large batteries can overwhelm an application… who wants a noticeable battery protruding at the entrance to their home? And harvested energy sources like solar aren't always available.

We are surrounded by an invisible intelligence, waiting to be freed. But first we must figure out how to give power to that intelligence.

Figure 1. IoT devices, including smart kitchen appliances, often need to be efficiently powered to be effective.

What is Invisible Intelligence?

Many interesting internet of things (IoT) applications adhere to principles of invisible intelligence. Inspired by the notion that the best technology fades away, invisible intelligence speaks a little more specifically to IoT applications.

The first principle of invisible intelligence is that when things are enabled by technology to expose that intelligence, it should not fundamentally change the way we interact with them. We still set the dial on a smart thermostat, but the thermostat starts to learn our preferences based on a number of conditions. We still glance at our wrist to tell the time, but we are presented with other meaningful data that we can explore further if we wish.

The second principle of invisible intelligence is that we shouldn't notice that the smart device is fundamentally different from its 'dumb' counterpart; this is how it is invisible. A smart bed with a noticeable hum as heating coils are activated defeats the purpose of the bed. A window sensor that needs a battery so large it blocks part of the light from the outside defeats the purpose of the window. If the motor on the automatic blinds can be obviously spotted, it defeats the effect of a flexible view.

The third principle of invisible intelligence is that the data needs to have value. Invisibly intelligent things will inherently be more expensive than their dumb counterparts. A smart brick might be easy to use and won't change how you interact with or think about bricks. But will buying more expensive bricks that will send you a text message when they fall off your house result in a good return on investment (ROI)?

Let's consider the three principles of invisible intelligence and how they pertain to our original question: how do we power the IoT?

- Our interaction: batteries or other harvesting mechanisms can't get in the way of your use of a device. Need to turn a hand crank to operate your smart thermostat? Battery on your smart watch weighing down your arm? Solar panel blocking access to your smart lock? We shouldn't just consider the physical impediments that these power supplies present. After all, battery charging is also not generally part of our interaction with most devices. While some battery re-charging may be necessary, designers should consider our natural interaction with the devices: a smart doorbell that needs daily recharging won't become an indispensable part of our lives.
- The device's appearance: here we consider aesthetics and practicality. Consider a view through a window being blocked by the battery of a smart window sensor: this will result in the window sensor being discarded. A smart ring that you are constantly aware of because a battery made it bulky won't be used once the novelty has worn out.

- The data's value: any IoT device has an inherent ROI, and as in any business, the 'return' needs to outweigh the 'investment.' Exotic harvesting mechanisms can drive up cost, making the 'I' outweigh the 'R'. Battery replacement also increases 'I,' not just in battery cost but in less tangible measurements like customer happiness.

Wearables Lead the Way

The IoT is not a moment or a market, but describes the trend of more and more connected devices and the technology enabling those devices. We already see IoT devices in mass deployment; smart meters and wearable fitness devices are a couple of popular examples. And in the case of the latter, we already see products achieving success while tackling these power questions and successfully exhibiting 'invisible intelligence.'

Commercially successful wearable devices, such as fitness watches, adhere to the three principles above. Our normal interaction with a fitness watch is no different than our interaction with a traditional watch—we still get the information we want at a glance, although we may need to learn more complex commands to access additional data and features. Many fitness watches do need a daily recharge, but it isn't that far from our 'normal interaction' with a watch—we generally remove it at night, and asking for a recharge (when we already do it for our phones) doesn't really disrupt our interaction.

Fitness watches also maintain the appearance of regular watches to a large extent. Thanks to advances in low-power technology, they are able to deliver more features and data without increasing their size beyond that of conventional watches.

Fitness watches also continue to provide valuable data. Years ago it was enough to tell a runner or biker how far they went, but now they count calories, analyze multiple health parameters, recognize many different kinds of exercise, and integrate with their smartphones for instant notifications.

In addition to adhering to the principles of invisible intelligence, fitness watches also demonstrate another trend in the IoT: the ever-increasing thirst for data. Products that only had GPS a few years ago now have multiple optical sensors for heart rate and pulse oximetry, motion sensors to fill in the gaps where GPS won't work and identify other exercises, altimeters, and more. They also come with more ways to deliver data: wireless interfaces and colorful graphical displays have replaced proprietary wired connections and monochrome text displays.

Wearable applications like fitness watches may lead the way, but we see lots of other connected devices adding more sensors to help provide more data. But more sensors and more connectivity and more algorithms mean more power, and bigger batteries or more frequent recharges are never a good answer. How can applications manage their power budget while appearing to do the impossible: provide more data while extending the life of the device?

Power and the Real World

We generally think about three key capabilities for IoT applications: sensing (and sometimes control), processing, and communicating. But often forgotten and left to consider for the end is power.

Many IoT devices can be plugged in—smart kitchen appliances or power tools often need wall power anyway. Power consumption may not be so much of an issue here, and this even opens up more communication options such as powerline communication. But the real promise of the IoT—that we can get valuable data from everywhere—can only be fulfilled if we can shed the power cords and get data from things that aren't tethered to an infinite power source.

Harvested energy is a great idea, but due to the nature of harvested energy it isn't widely practical except in niche use cases. Solar power could produce plenty of energy for many applications, but how many applications can take a day or a week off when the weather turns poor? And how many applications can take a break at night? You can fill those gaps with rechargeable batteries, but then you start to lose the value of going solar.

The fact is that most IoT applications are going to rely on batteries of some kind. Even with harvested energy sources there will still likely be a rechargeable battery, and many applications will depend on a primary battery and either assume the device should be discarded after its useful life or assume that the device is valuable enough to warrant a battery replacement.

To maximize lifetime of the device or time between charges, design engineers must make efficient use of the battery.

Delivering Power Efficiently

Most embedded applications are going to make use of several ICs, each needing power to function properly. In many cases, those ICs may want to see several different voltage supplies. You might have chips that want to run at 1.8V, at 3.3V, and 1.2V. You might see individual chips wanting multiple different supply rails. It doesn't make a lot of sense to have one battery for each one of those supply rails, so designers will need to figure out how to take a single battery (producing a single voltage) and turn it into all the supply rails that they need. The designer will also need to figure out how to deal with characteristics of real batteries: for example, over the life of the battery the voltage it produces will change, typically dropping as the battery discharges.

Simple, inexpensive linear regulators can be used to convert a battery voltage to necessary supply voltages. In fact, many ICs may integrate linear regulators to simplify external circuit design. Microcontrollers often use many different voltages internally. I/O pins might want to function at 3.3V, while core logic may want to function at 1.8V—in this case, the microcontroller could use a linear regulator so the designer only needs to supply the microcontroller with 3.3V, and it will internally regulate a 1.8V supply.

The real promise of the IoT comes when we can cut the power cords

While linear regulators can provide very low noise supplies which could be critical for more sensitive analog circuitry, they can be highly inefficient because they waste a significant amount of power to convert from one voltage to another.

Switched-mode power supplies (SMPS) are generally designed to deliver power more efficiently to an embedded application. These designs are typically more complex and may carry more cost. If not properly managed, the noise from the switching elements can disturb other circuitry. However, the benefit is that instead of linearly burning excess power in the conversion from one voltage to another, switched-mode power supplies often convert voltages with efficiencies of 90% or more.

What does this mean? Consider a microcontroller that runs at 1.8V and consumes 100µA running at 1MHz in a system where the supply battery is at 3.3V. With a linear regulator, a proportionate amount of current is spent regulating from 3.3V to 1.8V, so while the microcontroller is consuming 180µW, the linear regulator is consuming another 150µW.

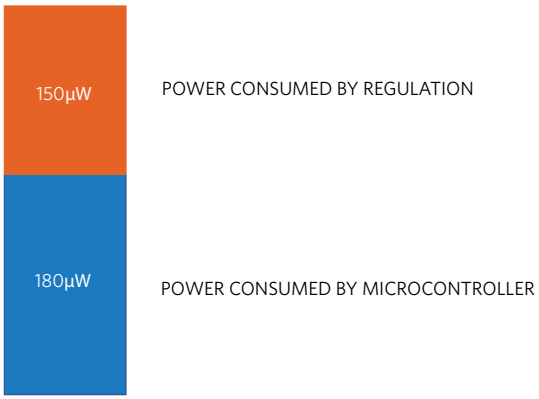

Figure 2. Power consumption by linear regulator.

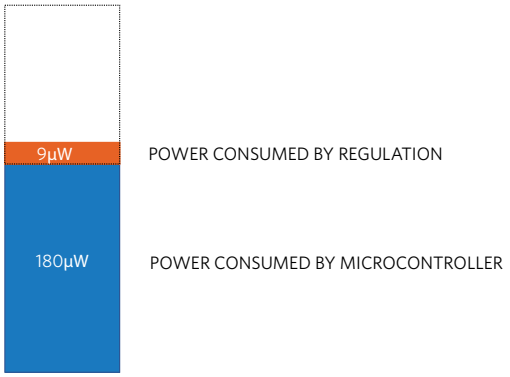

Figure 3. Power consumption by switching regulator.

Now consider the same situation with an SMPS with 95% efficiency. In this case the microcontroller is still consuming 180µW, but the power supply only consumes another 9µW for a total power consumption of 189µW instead of 330µW

Consider POWER Instead of CURRENT

As a society we talk a lot about saving power. We might even pay more for appliances with lower power consumption. The electric utilities bill us based on the amount of power we use. No one is really concerned with how much current the devices consume, because the measure is meaningless without knowing at what voltage the devices are running. Why then do microcontrollers use microamps per Megahertz (µA/MHz) as a common benchmark for efficiency? A more fitting benchmark would be microwatts per Megahertz (µW/MHz).

Consider the following two microcontrollers: Micro A runs at 60µA/MHz off a 3.3V supply, while Micro B runs at 100µA/MHz off a 1.2V supply.

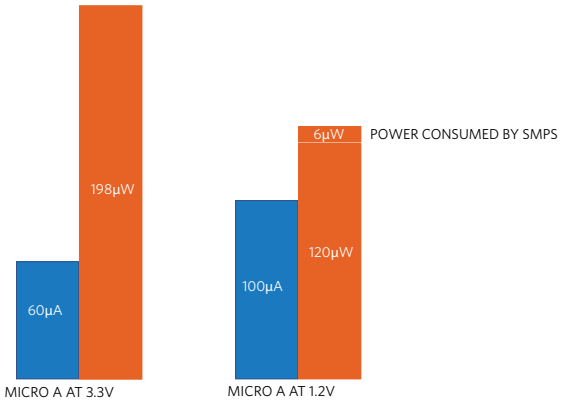

Figure 4. Power = Current TIMES Voltage.

We hear repeatedly that Micro A is better because it consumes less current, only 60µA/MHz instead of 100µA/MHz. But if we think about power (and consider the micro running at 1MHz), then even with a 5% efficiency reduction due to an SMPS delivering the proper rail, Micro B consumes far less power than Micro A.

Choosing a Low-Power Microcontroller

Some products can be chosen for power consumption based on a couple of lines in the datasheet (and a healthy dose of validation). But microcontrollers are trickier—as the brain of your embedded system, they can be a dominant consumer of your battery. And also like a brain, they are far too complex to choose based on one or two lines in a datasheet.

We've already talked about the "µW/MHz" benchmark with regards to supply voltage, but while this benchmark is attractive for its simplicity, it is not an accurate predictor of how well a microcontroller will perform in your application. Your application will likely need to make use of peripherals and run interesting application code, while the "µW/MHz" benchmark tends to be run while the microcontroller core is running a simple code loop. The "µW/MHz" benchmark can be used as perhaps a first filter—if one product is significantly worse than another, it is not likely to make up for it with further inspection. But if two products are relatively close (maybe even 25%-50% away from each other), it is probably worthwhile to dig a little deeper.

Low-power sleep modes are also important in determining the longevity of an application. In these modes, the microcontroller is not active, and the most power-hungry circuits in the chip are not drawing power. But even low-power sleep modes are complex, and many chips support multiple low-power sleep modes. There are many important factors to consider when making using of these low-power modes:

- How much memory is being retained? It can be important to retain SRAM in a low-power sleep mode: without

Don't choose a low-power microcontroller based only on the headlines in the datasheet

SRAM retention the microcontroller wouldn't be able to remember datalogs, operating system state, network state, intermediate algorithm results, or other data critical to the application. Simple applications may not need to remember data like this, but consider an application like an ultrasonic water meter: flow measurements are generally taken only a couple of times a second, and data only needs to be reported a few times an hour or a day. Compensation algorithms will want to look at larger amounts of data to increase system accuracy—if your microcontroller couldn't retain a decent amount of data, it would need to store it to an EEPROM or other nonvolatile storage before every sleep cycle.

This storage would burn significant power to store intermediate results that on their own aren't interesting for the end application. Or imagine a more complex application running an operating system—a larger SRAM retained in low-power modes means the microcontroller doesn't need to waste time and power re-initializing its operating system when it comes out of sleep mode. Note that retaining a bigger amount of SRAM generally will burn more sleep mode current than a smaller retained SRAM (provided other parameters stay the same).

- What is the wakeup time? Clocks in a microcontroller can consume a significant amount of power. In sleep modes, the higher frequency clocks tend to be turned off to save power. But to wake back up, those clocks need time to stabilize—during this time the microcontroller isn't doing useful work, but it is starting to consume more power. A faster wakeup minimizes the amount of power wasted transitioning between lower power modes and active modes. In addition, a fast wakeup time may be required by the application to properly respond to an event. To wake up to data presented on a serial interface, the microcontroller must wake up fast enough to properly recognize the incoming data bits.
- Do you want to sleepwalk? Some microcontrollers have the ability to do interesting things without the microcontroller being awake! Imagine an application that wants to wake up 100 times a second to read 16 bits of data from an SPI-connected analog-to-digital converter (ADC). The brute force approach would be to wake up the microcontroller, activate the SPI interface, read the data, write to memory, and go back to sleep. While the function is no doubt important to the application, it seems like a pretty menial task for an advanced 32-bit microcontroller that can figure out how to turn seemingly random noise into a medical diagnosis. Instead of engaging the microcontroller core, some microcontrollers offer options to handle routine tasks like this with a simple state machine—in these low-power modes, the high-speed clocks may still be activated, but the big microcontroller is kept asleep since we don't need its horsepower. A direct memory access (DMA) is a simple example of this (it can move data from one location to another without the microcontroller being involved), but

more intelligent options are available to do programmatic actions like "read from SPI-connected ADC 100 times a second and store the data to SRAM," at a fraction of the power consumption of the main microcontroller.

- What is the duty cycle? Low-power sleep modes may not be that critical. Some applications may want to be "always on" for example, if they are constantly listening for audio or reading sensors at an extremely high frequency. The "duty-cycle" refers to how much of the time the application or microcontroller is active and doing interesting work. Consider the following two scenarios with a microcontroller that consumes 10mW of power when running and 3µW when sleeping: scenario 1 is a 1% duty cycle (the microcontroller is active 1% of the time) and scenario 2 is a 0.01% duty cycle. In scenario 1, the average power consumption is 103µW, and in scenario 2, the average power consumption is 4µW. Note that in either case a 1µW reduction (or increase) in sleep-mode power will reduce (or increase) the average power by about 1µW. So in scenario 1, the active-mode power is clearly more critical, while in scenario 2 the sleep-mode power is clearly more critical.

How you tap into low-power modes depends on your design

Summary

With the breadth of potential IoT applications, there will never be a single power parameter that is the most critical for a microcontroller to achieve, nor will there be an IoT microcontroller suitable for all applications. Designers have a challenging task of determining not just the best-performing microcontroller for their current requirements, but of also determining the amount of flexibility they might need for potential in-field upgrades and changes in the application use case that might change their assumptions on power.

In the end, power isn't the only critical parameter for building the world of invisible intelligence. Without security and integration, the things around us won't be able to integrate intelligence and deliver trusted information. However, without sufficient power performance, IoT devices won't last long enough to provide a positive ROI.

More Resources

Learn more about a new breed of low-power microcontrollers built for the IoT:
Meet DARWIN

Also, read our related white papers:
Adding Intelligence to the Next Generation of Smart Devices

Securing the Next Generation of Smart Devices

For more information, visit:
www.maximintegrated.com